ELEC 1520- Programming for Electrical Engineering

Bit Manipulation LED

**Background**

Calculators, watches, and other electronic devices often rely on seven-segment displays with LEDS (light emitting diodes) for numerical output. To form a digit, some segments are turned on, while leaving the others off.

A picture containing electronics, keyboard

Description automatically generated

Fill in the table below. Assume a 1 turns a segment on, and a zero turns a segment off. Example: To display the digit zero, all segments except G are turned on***. You do not need to turn in the table, only the code, which will use the results of your table. You must encode the table within your code using a map.***

Example: map <unsigned int, unsigned char> SegmentTable;

SegmentTable[4] = 0x66;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Segment Display Positions | | | | | | |  |
|  | Unused | G | F | E | D | C | B | A |  |
| 7SegmentDisplay Output | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Hexadecimal Representation of bits |
| 0 | 0 |  |  |  |  |  |  |  |  |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0x06 |
| 2 | 0 |  |  |  |  |  |  |  |  |
| 3 | 0 |  |  |  |  |  |  |  |  |
| 4 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0x66 |
| 5 | 0 |  |  |  |  |  |  |  |  |
| 6 | 0 |  |  |  |  |  |  |  |  |
| 7 | 0 |  |  |  |  |  |  |  |  |
| 8 | 0 |  |  |  |  |  |  |  |  |
| 9 | 0 |  |  |  |  |  |  |  |  |
| A | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0x77 |
| b | 0 |  |  |  |  |  |  |  |  |
| C | 0 |  |  |  |  |  |  |  |  |
| d | 0 |  |  |  |  |  |  |  |  |
| E | 0 |  |  |  |  |  |  |  |  |
| F | 0 |  |  |  |  |  |  |  |  |

Use the code provided to generate the output of using 6 seven segment displays to generate the time of day on the clock. The clock display has 6 seven segment display controlled by one (unsigned long) 64-bit control variable that has 6 designated 7-segment LEDS positions (7-bits each) using the following arrangement.

Graphical user interface, application

Description automatically generated

For a 64-bit control system, the designated bit fields use the following A63..A0 meaning:

Graphical user interface, text, application

Description automatically generated

// Assume clock time of HH:MM:SS

// SS are bits 13..0

// MM are bits 27..14

// HH are bits 41..28

// bits 63..42 are 0

Write a C++ program that displays hexadecimal digits or decimal digits, based on the user’s input at the command line using . Prompt the user to input a character.

./main 12:34:56

This starts the hours at 12, minutes at 34, and seconds at 56. The program is designed to use a timer to increment every second:

TIMER

--------------------------

| 12 hrs | 34 min | 56 sec |

--------------------------

>0000000000000000000000000000000000000000000000000000000000000000

To stop the timer/close, type: CTRL+c on repl.it

The program needs to take the input from argv[1] and generate a string and determine how to correctly process the different digits.

string time\_of\_day = string(argv[1]);

Conver the time\_of\_day into hours, minutes, and seconds to pass to the function timer().

Then, within the function generateClockControl(), convert hours, minutes, seconds, into a form of HH:MM:SS that can set the 64-bit respective fields.

Also, there there are certain times that represent interesting processing conditions. For example in the case of 9:02:00, the top hours field is empty.

./main 9:02:00

The display for this will be 09:02:00

The input does not need to detect invalid inputs to the program. For example ./main 12:123das:12

Steps to complete the assignment:

**Step 1:**

Convert the string of the commandline argument using streaming strings and replace() into 3 integers: hours, minutes, seconds

**Step 2:**

Build a C++ map initialization with the input value keys 0..9 (that represent the numbers of a clock) that map to the hexadecimal encoding of 7-segment display in the computer display system.

map<unsigned int, unsigned char> SegmentTable

**Step 3:**

Finish the coding of “generateClockControl” which creates each binary encoding of the clock: h1, h0, m1, m0, s1, s0 into a formatted 64-bit variable.

Discussion: If given hours, minutes, seconds as 12, 34, 56, how to determine h1 and h0?

unsigned long generateClockControl(int hours, int minutes, int seconds)

{

unsigned long h1,h0; // 12 h1=1 h0=2

unsigned long m1,m0; // 34 m1=3 m0=4

unsigned long s1,s0; // 56 s1=5 s0=6

unsigned long output;

return output;

}

// char (byte) 8 bits

// short (2 bytes) 16 bits

// int (4 bytes) 32 bits)

// long (8 bytes) 64 bits. 32 2-bit 1 64-bit. 8 16 bit

// 64 bits [17 bits are temperature, 13 bits are pressure, 1 yes/no]